RI 9339

RI 9339

REPORT OF INVESTIGATIONS/1991

Genetic Algorithm Applied to Least Squares Curve Fitting

By C. L. Karr, D. A. Stanley, and B. J. Scheiner

UNITED STATES DEPARTMENT OF THE INTERIOR

BUREAU OF MINES

Report of Investigations 9339

Genetic Algorithm Applied to Least Squares Curve Fitting

By C. L. Karr, D. A. Stanley, and B. J. Scheiner

UNITED STATES DEPARTMENT OF THE INTERIOR
Manuel Lujan, Jr., Secretary

BUREAU OF MINES
T S Ary, Director

This report is based upon work done under an agreement between the University of Alabama and the Bureau of Mines.

Library of Congress Cataloging in Publication Data:

Karr, C. L. (Charles L.)
Genetic algorithm applied to least squares curve fitting / C. L. Karr, D. A. Stanley, and B. J. Scheiner.

p. cm. — (Report of investigations; 9339)

Includes bibliographical references.

1. Combinatorial optimization—Data processing. 2. Curve fitting—Data processing. 3. Least squares—Data processing. 4. Algorithms. I. Stanley, D. A. (Donald A.) II. Scheiner, Bernard J. III. Title. IV. Series: Report of investigations (United States. Bureau of Mines); 9339.

TN23.U43 [QA402.5] 622 s—dc20 [519.3'0285] 90-2602 CIP

CONTENTS

UNIT OF MEASURE ABBREVIATIONS USED IN THIS REPORT

cP	centipoise	pct	percent
g/Kg	gram per kilogram	s	second

GENETIC ALGORITHM APPLIED TO LEAST SQUARES CURVE FITTING

By C. L. Karr,[1] D. A. Stanley,[2] and B. J. Scheiner[3]

ABSTRACT

The U.S. Bureau of Mines is currently investigating the use of genetic algorithms (GA's) for solving optimization problems. This computer search technique, based on the mechanics of natural genetics, was used to perform simple least squares curve fitting. Three examples are presented in which a GA manipulates binary coded strings to produce near-optimal solutions to least squares curve-fitting problems after having viewed only a small portion of the search space. The examples include fitting data to the equation of a line, the equation of a parabola, and the Ree-Eyring equation.

[1]Mechanical engineer.
[2]Supervisory research chemist.
[3]Supervisory metallurgist.
Tuscaloosa Research Center, U.S. Bureau of Mines, Tuscaloosa, AL.

INTRODUCTION

Curve fitting plays an important role in the analysis, interpretation, and correlation of experimental data. Although there are many approaches to curve fitting, the method of least squares can be applied directly to problems involving linear forms with undetermined constants. However, the conventional least squares method of curve fitting does have limitations; nonlinear forms and forms for which no derivative information exists present problems. These limitations may be overcome by incorporating a GA.

GA's are search algorithms based on the mechanics of natural genetics (*6*).[4] They efficiently exploit old knowledge contained in a population of solutions to generate new and improved solutions. GA's have been used in a variety of problems (*1-2, 4-5, 7*) where they have been shown to converge rapidly to near-optimal solutions after having sampled but a small fraction of the search space.

In this U.S. Bureau of Mines report, a simple GA is applied to three least squares curve-fitting problems. Although the problems have been effectively solved using more conventional techniques, they serve as a useful check on the principle of using a GA for solving curve-fitting problems. Furthermore, the method of curve fitting data using a GA is easily extended to more complex problems that are difficult for the conventional least squares technique.

HOW GENETIC ALGORITHMS ARE DIFFERENT

GA's are broadly applicable, efficient search algorithms based on the mechanics of natural genetics. They imitate nature with their Darwinian survival-of-the-fittest approach. This approach allows GA's to speculate on new points in the search space with expected improved performance by exploiting historical information. Because GA's imitate nature, they exhibit some fundamental differences from more conventional search techniques. GA's differ from more conventional search techniques in three ways:

1. GA's work with bit strings that represent entire parameter sets, whereas most methods manipulate individual parameters independently.
2. GA's consider many points in the search space in each iteration.
3. GA's use random choice to guide their search for which they require no derivative information.

GA's require the natural parameter set of the problem to be coded as a finite string of bits. The parameter sets in this study are coded as strings of zeros and ones. For example, the two constants needed to define a line of the form $y = C_1x + C_2$ are quite easily represented as a binary string. Eleven bits are allotted for defining each constant. The first bit position is devoted to the sign of C_1; i.e., when the value is zero C_1 is positive, and when the value is one C_1 is negative. The next 10 bits, positions 2 through 11, are interpreted as a binary number (1001010111 is the binary number 599). This value is mapped linearly between some user-determined minimum (C_{min}) and maximum (C_{max}) values according to the following:

$$C_1 = C_{min} + \frac{\text{binrep}}{(2^{\ell} - 1)} (C_{max} - C_{min}), \qquad (1)$$

where binrep = the integer value represented by an ℓ bit string. The values of C_{max} and C_{min} in a given problem are selected by the user based on personal knowledge of the problem. If necessary, a rapidly converging, course optimization method may be used for selecting the limiting values. This same form is used to represent C_2, and the two 11-bit strings are concatenated to form a single 22-bit string representing the entire parameter set (C_1 and C_2). In other problems, the creation of appropriate finite string codings may require more complicated mappings, but with a little creativity the possibilities are endless.

Since GA's work directly with a coding of the parameter set and not the parameters themselves, they are difficult to fool because they are not dependent upon continuity of the parameter space. GA's consider many points in the search space simultaneously and therefore have a reduced chance of converging to local maximums. In most conventional search techniques, a decision rule governs the movement from one point to the next. These methods can be dangerous in multimodal (many peaked) search spaces because they can converge to local maximums. However, GA's generate entire populations of points (coded strings), test each point individually, and combine qualities from existing points to form a new population containing improved points. Aside from producing a more global search, the GA's simultaneous consideration of many

[4]Italic numbers in parentheses refer to items in the list of references at the end of this report.

points makes them highly adaptable to parallel machines since the evaluation of each point is an independent process.

A GA requires only information concerning the quality of the solution produced by each parameter set (objective function values). This is contrary to many optimization methods that require derivative information or, worse yet, complete knowledge of the problem structure and parameters. Since GA's do not require such problem-specific information, they are more flexible than most search methods.

Last, GA's differ from more typical search techniques in that they use random choice to guide their search. Although chance is used to define their decision rules, GA's are not random walks through the search space. They use chance efficiently in their exploitation of prior knowledge to rapidly locate near-optimal solutions.

SIMPLE GENETIC ALGORITHM

Although some GA's have become quite complex, good results can be rapidly achieved with relatively simple GA's. The simple GA used in this study consists solely of reproduction, crossover, and mutation—operations basic to all GA's. As will be seen, these operations are easily implemented on a computer.

Before examining the individual operations, consider the overall processing of a GA. An initial population of n strings each of length ℓ are generated at random. Keep in mind that each string represents one possible solution to the problem at hand, one possible combination of the input parameters. Each string is decoded yielding the actual parameters. The parameter set represented by each string is sent to a numerical model of the problem, a solution based on the input parameters is returned, and the string is assigned a fitness value, which is simply a nonnegative measure of the quality of the string's solution. The assignment of fitness values is problem dependent; they are defined by the user to represent the quality of a string. This fitness is then used to direct the application of the three operations, which produce a new population of strings (a new generation). Hopefully, this new generation will contain better solutions to the problem. The new population of strings is again individually decoded, evaluated, and transformed into a subsequent generation using the basic operations. This relatively simple process continues until convergence within a population is achieved.

Reproduction is simply a process by which strings with large fitness values, good solutions to the problem at hand, receive correspondingly many copies in the new population. In this study, use is made of a particular type of reproduction, expected number control. This form of reproduction makes a certain number of copies, num_i, of a string in accordance with the equation:

$$num_i = \frac{f_i}{f'}, \qquad (2)$$

where num_i = the number of copies of the string in the next generation,

f_i = the fitness of the individual string in the current generation,

and f' = the average fitness of the current generation.

(A mechanism for handling roundoff and assuring a constant population size is required.) Thus, reproduction is the survival-of-the-fittest aspect of the GA. The best strings receive more copies in subsequent generations so that their desirable traits may be passed onto their offspring.

Crossover affords a means for strings to mix and match their desirable qualities through a random process. After reproduction, simple crossover proceeds in three steps. First, two newly reproduced strings are selected from the strings created by previous selection. Second, a position along the string is selected at random. This is shown as follows, where two binary coded strings X and Y of length six are nested for crossover:

$$\begin{aligned} X &= \overline{1\,1\,0} \mid 1\,0\,1 \\ Y &= 0\,0\,1 \mid \underline{0\,1\,1}. \end{aligned}$$

Notice how crossing site 3 has been selected in this particular example through random choice, although any of the other four positions could have been selected just as easily. The third step is to exchange all characters following the crossing site. The two new strings following this example cross are shown X' and Y':

$$\begin{aligned} X' &= 1\,1\,0\,0\,1\,1 \\ Y' &= 0\,0\,1\,1\,0\,1. \end{aligned}$$

String X' is made up of the first part of string X and the tail of string Y. Likewise, string Y' is made up of the first part of string Y and the tail of string X. Although crossover has a random element, it should not be thought of as a random walk through the search space. It is an effective means of exchanging information and combining portions of high-quality solutions.

The mechanics of the reproduction and crossover operations are quite simple; they involve nothing more than making copies of strings and exchanging portions of strings. However, reproduction and crossover together give GA's much of their power.

Reproduction and crossover give GA's the majority of their searching power, but the third operation, mutation, enhances the ability of the GA to find near-optimal solutions. Mutation is the occasional alteration of a value at a particular string position. Its purpose is to serve as an insurance policy; it insures against the loss of a particular piece of information. A generation may be created that is void of a particular character at a given string position. For example, a generation may exist that does not have a one in the third string position when a one at the third position may be critical to obtaining a quality solution. Neither reproduction nor crossover will ever produce a one in this third position in subsequent generations. Mutation, however, allows for the possibility of a zero in the third position to be changed to a one. Thus, the critical piece of information can be reinstated into the population. Although mutation is a vital part of any GA, it occurs with a small probability (on the order of one mutation per thousand string positions).

This has been a brief overview of a simple GA. For details of the processing power and the convergence properties of GA's, reference should be made to Holland (*6*).

GENETIC ALGORITHM APPLIED TO LEAST SQUARES CURVE-FITTING PROBLEMS

In this section, a simple GA is used to solve three least squares curve-fitting problems. Although the problems investigated have been well solved by other methods, they serve as a demonstration of the GA's versatility and power in the area of curve fitting data.

Three examples are presented in which results produced by the GA are compared with the optimum results of the conventional calculus based least-squares technique. The first example is a line of the form $y = C_1x + C_2$. This is a simple example presented to illustrate the basic premise of using a GA for curve fitting. The second and third examples are more complex problems that are of immediate interest to the Bureau. In the second example, data is fitted to an equation of the form

$$A^{C_1} = C_2C + C_3. \quad (3)$$

In this equation,

	A =	the amount of polymer required for dewatering (*8*) (the dependent variable),
	C =	the polymer concentration (the independent variable),
	C_1 =	an undetermined constant which causes the curve to be linear,
and	C_2 and C_3 =	constants defining the line.

The third example involves fitting data to an equation of the form $y = C_1 + C_2\sinh^{-1}(C_3x) / (C_3x)$. This form was selected because fitting data to this equation allows for the solution of the Ree-Eyring equation, which is used to estimate non-Newtonian viscosity components. These three problems by no means represent all the existing curve fitting problems, but they are exemplary of problems commonly faced in data handling.

PRELIMINARY CONSIDERATIONS

Before considering the three specific problems, two points concerning the application of the GA must be addressed. There are basically two decisions to be made when applying a GA to any given problem: (1) how to code the parameters of the problem as a finite string, and (2) how to evaluate the merit of each string (each parameter set).

The method for coding the strings has been discussed previously, so consider the question of how to evaluate each string. In least squares curve-fitting problems, the objective is to minimize the sum of the squares of the distances between a curve of a given form and the data points. Thus, if y is the ordinate of one of the data points, and y' is the ordinate of the corresponding point on the theoretical curve, the objective of least squares curve fitting is to make $(y - y')^2$ a minimum. This square of the error, which is to be minimized, affords a good measure of the quality of the solution. However, the GA seeks to maximize the fitness. Thus, the minimization problem must be transformed into a maximization problem. To accomplish this transformation, the error is simply subtracted from a large positive constant (Const) so that

$$f = \text{Const} - (y - y')^2 \qquad (4)$$

yields a maximization problem.

The selection of an appropriate coding and the determination of a fitness representation are the only aspects of a GA that are problem specific. Once these two decisions have been made, every problem is the same to a GA. The GA generates populations of strings, evaluates them, and uses these evaluations to produce subsequent generations of more highly fit strings. Thus, the GA is a very flexible search method.

The simple GA described earlier has been programmed in the Pascal programming language by modifying the code found in Goldberg (*3*). The GA is run in these studies with the following parameters:

$$\text{popsize} = 100$$

$$P_{\text{crossover}} = 0.65$$

$$P_{\text{mutation}} = \frac{1}{\text{popsize}}.$$

These values are consistent with De Jong's suggestions (*1*) for moderate population size, high crossover probability, and low mutation probability.

EXAMPLE 1—LINE

This problem involves the fitting of data to a line having the form $y = C_1x + C_2$. Both C_1 and C_2 must be selected to minimize the square of the distance between the points and the curve. Six data points were selected to be fitted and are as follows:

X	Y
0	0
1	1
2	2
3	3
4	4
10	10

These points fall directly on the straight line represented by the constants $C_1 = 1$ and $C = 0$. These values are readily found using the conventional least squares technique.

Figure 1 shows the best-of-generation error for two independent runs (different starting populations are provided because results depend on a random seed) as it decreases with successive generations. Recall that each generation represents the creation of n = 100 strings where n * $P_{\text{crossover}}$ = 100 * 0.65 = 65 of the strings are new. It is interesting that near-optimal results are obtained after only six generations (approximately 400 new function evaluations).

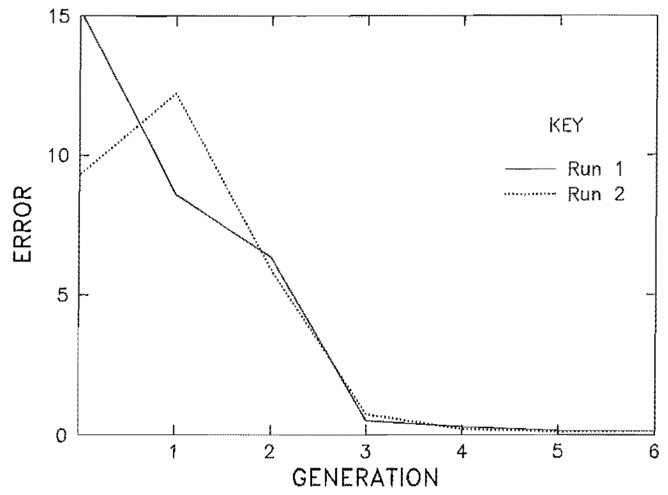

Figure 1.—Best-of-generation error versus generation number for example 1.

For this example of fitting the data to the equation of a line, near-optimal results are obtained even though the size of the search space is large (2^{22} = 4.2 x 10^6 points), and the number of points explored is small (400). This is exactly the type of performance the GA gives; it finds near-optimal solutions after having viewed only small portions of the search space.

To get a feel for the results produced by the GA, consider the best solution from run 2. These results are compared with the optimum results as found using a conventional least squares curve-fitting technique in figure 2. The constants selected by the GA were C_1 = 0.9688 and C_2 = 0, which produced a squared error of 0.127 versus the optimum value of 0. C_1 and C_2 were allowed to vary from -32.0 to 32.0.

EXAMPLE 2—POLYMER REQUIRED FOR DEWATERING VERSUS CONCENTRATION

The Bureau is investigating a method of dewatering flocculated fine-particle waste that involves treating the waste with a high-molecular-weight polymer such as polyethylene oxide (PEO). Stanley and Scheiner (*9*) found an empirical relationship between the amount of polymer required for dewatering the clay and the polymer concentration. The equation is of the form

$$A^{C_1} = C_2C + C_3, \qquad (5)$$

where A = amount of polymer (PEO) required in gram per kilogram of clay to dewater a particular clay (the dependent variable),

C = concentration, in weight percent, of the polymer solution used for dewatering (the independent variable),

C_1 = constant that determines the curvature of the plot,

and C_2 and C_3 = constants defining the line.

To establish the correlation, A is raised to the C_1 power and plotted versus C. This is illustrated for an unexchanged attapulgite clay dewatered with 0.05 pct PEO solution. The data points are plotted for values of C_1 of 1.7, 2.1, and 2.5 as shown in figure 3. A unique straight line is obtained as C_1 approaches 2.1. Prior to this study, the selection of the constants had been a trial-and-error process. The use of a GA in fitting data to an equation of the given form shows its effectiveness in solving complex curve-fitting problems.

The addition of a third constant makes this example more difficult than the line because the size of the search space has increased significantly (from 2^{22} to 2^{33} points). Parameter sets are coded in the same manner as those of the line, except the strings representing the parameters now have a length of 33; the first 11 bits represent C_1, the second 11 bits represent C_2, and the third 11 bits represent C_3. The constants are once again determined based on the minimization of the square of the distance between the points and the curve. C_1, C_2, and C_3 ranged from -32.0 to 32.0.

Three data points were fitted for each of two clays: aluminum-exchanged and potassium-exchanged clays. Figure 4 shows the best-of-generation error for two independent runs as it decreases with successive generations. As in the previous example, the results of the GA quickly converge to near-optimal solutions. The results produced by the GA are compared with those of Stanley and Scheiner's (9) trial-and-error method in figure 5. The constants produced by the GA are compared with those of Stanley and Scheiner in table 1.

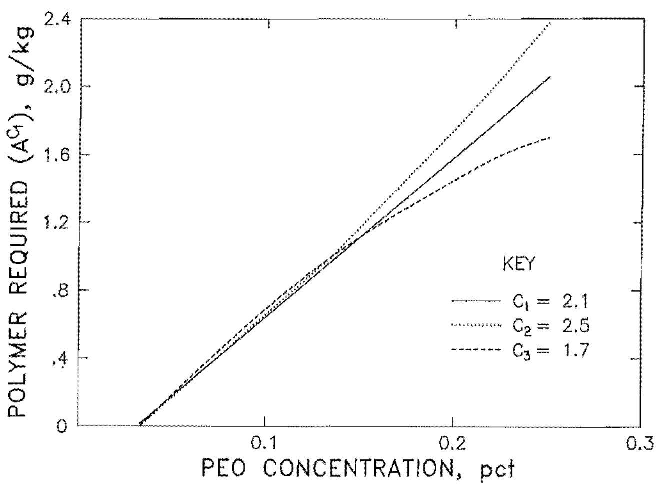

Figure 3.—Trial-and-error process.

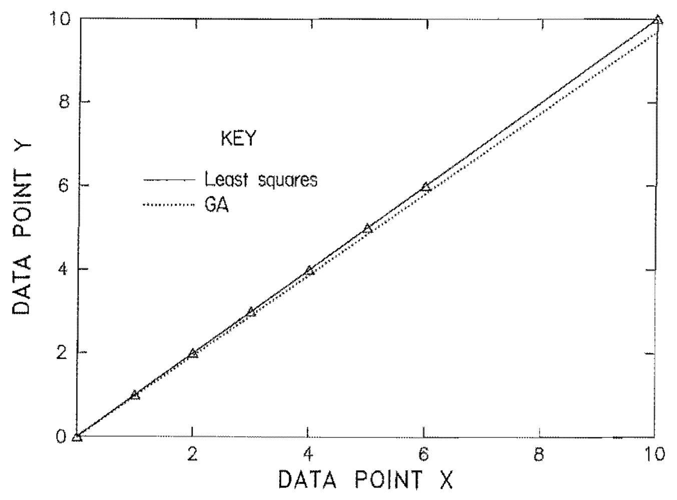

Figure 2.—GA-predicted curve fit for example 1.

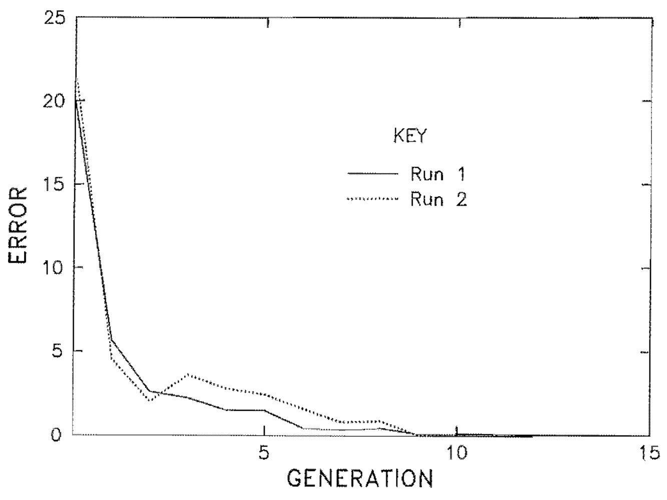

Figure 4.—Best-of-generation error versus generation number for example 2.

Table 1.—Comparison of trial-and-error and GA constants

	K^+	Al^{3+}
Trial and error (9):		
C_1	1.40	2.00
C_2	16.39	7.20
C_3	0	0
GA:		
C_1	1.40	2.00
C_2	16.280	7.063
C_3	-.031	.031

EXAMPLE 3—REE-EYRING EQUATION

The Ree-Eyring equation allows for the estimation of the non-Newtonian components of viscosity and is of interest to the Bureau in the area of dewatering phosphatic clay wastes. The Ree-Eyring equation may be written as

$$y = C_1 + C_2 \sinh^{-1}(C_3 x) / (C_3 x), \tag{6}$$

where y = the viscosity (the dependent variable),

x = the shear rate (the independent variable),

and C_1, C_2, and C_3 = the constants to be determined.

The process of fitting the data using a GA is the same as in the previous example. The parameter set (C_1, C_2, and C_3) is again represented as a 33-bit string of zeros and ones and is determined by minimizing the square of the distance between the data points and the curve.

Stanley, Webb, and Scheiner (10) used a transformation method to solve the Ree-Eyring equation for a sodium ion-exchanged clay. The data used in that report were also used here, and the results of the curve produced using the GA are compared with those previously obtained. Once again the GA quickly converged to a quality solution. Figure 6 shows the best-of-generation error for two independent runs as it decreases with successive generations. The smallest error produced by the GA calculated curve (1.57) is less than that of the curve calculated by the transformation method used by Stanley, Webb, and Scheiner (12.55). Figure 7 shows how well the curve calculated by the GA (C_1 = 168.83, C_2 = 34.20, and C_3 = 0.043) fits the data. C_1, C_2, and C_3 ranged between -200.0 and 200.0.

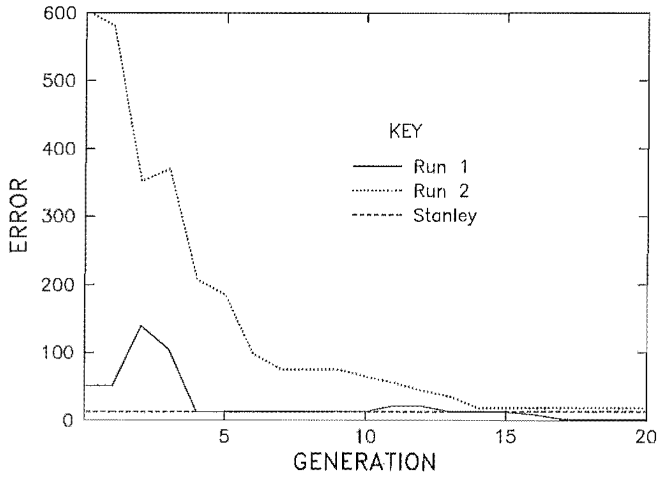

Figure 6.—Best-of-generation error versus generation number for example 3.

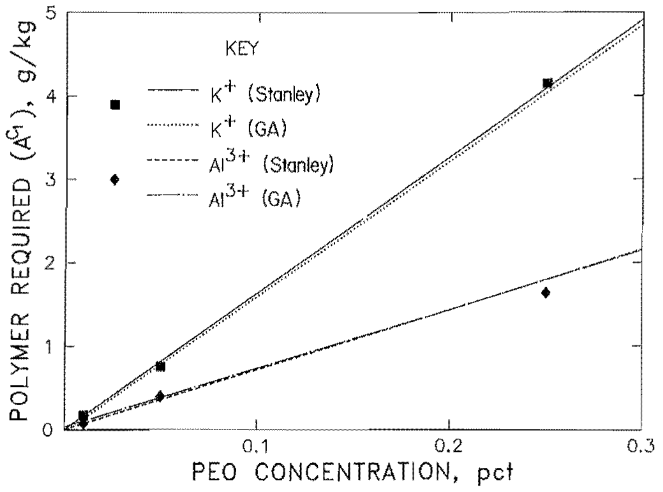

Figure 5.—GA-predicted curve fit for example 2.

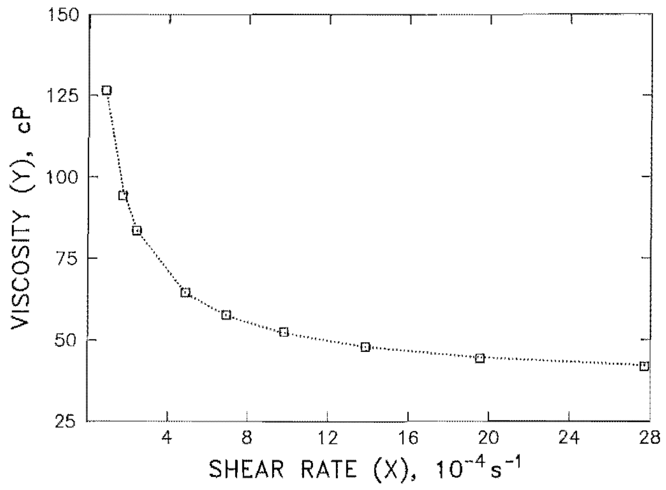

Figure 7.—GA-predicted curve fit for example 3.

CONCLUSIONS

Genetic algorithms are search algorithms that are able to locate near-optimal solutions after having sampled only small portions of the search space. They are flexible enough to be effective in a wide range of problems, well suited to parallel computers, and easily extended to multiple peak optimization problems.

In this report, a simple GA made up of reproduction, crossover, and mutation was used to perform least squares curve fitting. Three curve-fitting problems were presented in which a GA rapidly converged to near-optimal solutions. The solutions were comparable to, or better than, solutions determined by more commonly used methods. Although they require some knowledge of the problem, GA's are readily adaptable to least squares curve-fitting problems.

REFERENCES

1. De Jong, K. A. Analysis of the Behavior of a Class of Genetic Adaptive Systems. Dissertation Abstr. Int., v. 36, No. 10, 1975, p. 5140B.

2. Goldberg, D. E. Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning. Ph.D. Thesis, Univ. MI, Ann Arbor, MI, 1983, 227 pp.

3. ____. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Co., 1989, 432 pp.

4. Goldberg, D. E., and R. L. Lingle. Alleles, Loci, and the Travelling Salesman Problem. Paper in Proceedings of an International Conference on Genetic Algorithms and Their Applications, ed. by J. J. Grefenstette. Carnegie-Mellon Univ., Pittsburgh, PA, 1985, pp. 154-159.

5. Goldberg, D. E., and M. P. Samtani. Engineering Optimization Via Genetic Algorithm. Paper in Proceedings of Ninth Conference on Electronic Computation. Univ. AL Birmingham, Birmingham, AL, 1986, pp. 471-482.

6. Holland, J. H. Adaptation in Natural and Artificial Systems. Univ. MI Press, 1975, 183 pp.

7. Karr, C. L., and D. E. Goldberg. Genetic Algorithms in Mineral Processing and Machine Learning. Paper in Proceedings of the Artificial Intelligence in Mineral and Material Technology Conference. Univ. AL, Tuscaloosa, AL, 1987, pp. 127-141.

8. Scheiner, B. J., A. G. Smelley, and D. A. Stanley. Dewatering of Mineral Waste Using the Flocculant Polyethylene Oxide. BuMines B 681, 1985, 18 pp.

9. Stanley, D. A., and B. J. Scheiner. Mechanically Induced Dewatering of Ion Exchanged Attapulgite Flocculated by a High Molecular Weight Polymer. Colloids and Surf., v. 14, 1985, pp. 151-159.

10. Stanley, D. A., S. W. Webb, and B. J. Scheiner. Rheology of Ion-Exchanged Montmorillonite Clays. BuMines RI 8895, 1986, 17 pp.